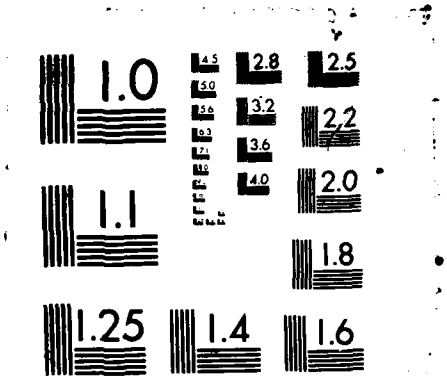
AD-A183 178 PROGRAMMING PRODUCTIVITY ENHANCEMENT BY THE USE OF APPLICATION GENERATORS (U) UNIVERSITY OF SOUTHERN CALIFORNIA LOS ANGELES DEPT OF COMPUTE E HOROWITZ

UNCLASSIFIED 31 DEC 86 53-4509-9601 AFOSR-TR-87-0940 F/G 12/5 NL

1/1



UNCLASSIFIED

DTIC FILE COPY

(2)

AD-A183 178

REPORT DOCUMENTATION PAGE

DTIC ELECTE AUG 12 1987 S D

Unclassified	1b RESTRICTIVE MARKINGS: N/A
2a SECURITY CLASSIFICATION AUTHORITY: N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT: Unlimited Distribution
2b DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A	
4 PERFORMING ORGANIZATION REPORT NUMBER(S): 53-4509-9601	5. MONITORING ORGANIZATION REPORT NUMBER(S): AFOSR-TR- 87-0940
6a NAME OF PERFORMING ORGANIZATION: Univ. of Southern California	6b OFFICE SYMBOL (If applicable):
6c ADDRESS (City, State and ZIP Code): University Park, Los Angeles, California 90089-1147	7a NAME OF MONITORING ORGANIZATION: Air Force Office of Scientific Research
8a NAME OF FUNDING/SPONSORING ORGANIZATION: AFOSR	8b OFFICE SYMBOL (If applicable):
8c ADDRESS (City, State and ZIP Code): Bolling AFB, Bldg. 410, Washington, D.C. 20332	7b ADDRESS (City, State and ZIP Code): same as 8c
	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: AFOSR ~~Grant Number~~ 82-0232

10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
61102F	2304	A3	

11. TITLE (Include Security Classification): Programming Productivity Enhancement by the Use of Application Generators

12. PERSONAL AUTHOR(S): Ellis Horowitz

13a TYPE OF REPORT	13b TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Final Report	FROM 6/1/82 TO 12/31/86	Dec. 31, 1986	9

16. SUPPLEMENTARY NOTATION: Project Title also known as "Investigating the Enhancement of Programming Languages with Multi-Media Capabilities"

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB GR	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This document is the final technical report for work under Air Force Office of Scientific Research Contract No. 82-0232. It is divided into several chapters. The first chapter deals with the specific research areas that were investigated and discusses the accomplishments for each. The areas of research are: (i) Application Generators, (ii) Office Information Systems, (iii) Software Engineering, and (iv) the ScritpWriter Software Development Environment. Chapter 2 reviews the progress of all people who have been supported under the grant. Three people have earned their Pd.D degrees while performing work supported by the grant. Several others are in various stages of completing their Ph.D. thesis, while others have graduated with M.S. degrees. Chapter 3 lists all of the publications that have been written under grant support.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS ☐	

22a NAME OF RESPONSIBLE INDIVIDUAL	22b TELEPHONE NUMBER (Include Area Code)	22c OFFICE SYMBOL
John P. Thomas ~~Ellis Horowitz~~ Ellis Horowitz	(213)743-6453 767-5026	NM

DD FORM 1473, 83 APR — EDITION OF 1 JAN 73 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

additional copies of a final report.

January 12, 1987

AFOSR-TR- 87-0940

Capt. John P. Thomas, Jr.
Department of the Air Force
Air Force Office of Scientific Research (AFSC)
Bolling Air Force Base, DC 20332-6448

Re: Final Scientific Report
AFOSR 82-0232

Dear Capt. Thomas:

Enclosed are 3 copies of my Final Scientific Report of above mentioned grant. The project was completed through this grant and I thank you for your assistance.

Sincerely yours,

Ellis Horowitz
Professor

cc: Randy Moore - NSF Grant Officer
Harriet Vigoren - USC Contract & Grant Administrator
Hilda Marti - USC Administrator Coordinator

Encl.

Final Technical Report for Grant No. AFOSR-82-0232 Programming Productivity Enhancement by the Use of Application Generators*

June 1, 1982 - December 31, 1986

Ellis Horowitz, Principal Investigator
Computer Science Department
University of Southern California
Los Angeles, California 90089
U.S.A.

December 31, 1986

*Project title also known as, "Investigating the Enhancement of Programming Languages with Multi-Media Capabilities."

Contents

Chapter 1

Major Research Accomplishments

1.1 Introduction

This document is the final technical report for work under Air Force Office of Scientific Research Contract No. 82-0232. It is divided into several chapters. The first chapter deals with the specific research areas that were investigated and discusses the accomplishments for each. The areas of research are: (i) Application Generators, (ii) Office Information Systems, (iii) Software Engineering, and (iv) the ScriptWriter Software Development Environment. The next chapter reviews the progress of all people who have been supported under the grant. Three people have earned their Ph.D. degrees while performing work supported by the grant. Several others are in various stages of completing their Ph.D. thesis, while others have graduated with M.S. degrees. Chapter 3 lists all of the publications that have been written under grant support.

1.2 Application Generators

When I originally wrote this research proposal, my focus was on the area of Application Generators. Systems such as RAMIS, NOMAD and FOCUS had all proven to be versatile at improving programmer productivity in the business sector. Their emphasis on nonprocedural programming and an interface to a database management system made them interesting candidates of study. My original intention was to examine the degree to which I could extend the Application Generator concepts to other worlds of programming. The first work to come out of this research was the paper [AppGenSur]. This paper made sense out of the various interpretations of nonprocedural programming and organized our understanding of their capabilities. We ex-

```
type PERSON_REL  is
     relation (key SS_NO)
      SS_NO : string(9);
      NAME : string(20);
      SEX : (F,M):
      SALARY : real;
 end relation;

  for P in PEOPLE where P.SEX = F  loop
      PEOPLE[P].SALARY := PEOPLE[P].SALARY * 1.1
 end loop;

PEOPLE := PEOPLE union NEW_GUY;
```

Figure 1.1: Example of AdaRel

amined closely several systems and then defined generalized features based upon a generic model.

Our second activity was to see if we could design application generator features into a general purpose programming language. We decided to use Ada as the starting point. To begin we designed an extension of Ada that permits the language to interface naturally with a database management system. We made this a true language extension as opposed to simply a collection of procedure calls. We added a type **relation** and provided a broad set of operators including **select**, **project** and **join**. In Figure 1.1 you see a small example that defines a relation, shows a for-loop that increases all people's salary in the relation by 10% and a final line that inserts a new person into the database. All of the features of AdaRel are fully defined in [AdaRel].

Our next step was to consider the report generation and graphic display features. This led us to consider the question of designing applications that deal with screens of information. We observed that conventional languages are wholly inadequate as their input/output capabilities are built around the concepts of characters and lines. This led us to further extend the AdaRel model so that it includes a new basic unit called **screen**. In figure 1.2 you see the definition of an AdaRel Screen. Briefly, a screen has two parts: a format part and an activation part. The format part permits the definition of the screen, while the activate part performs a set of user determined actions at run-time. A most important aspect of the work was to show how data of type **relation** can be merged with the **screen** concept. Given these concepts, we were able to write many application programs and show

```
screen  <screen name>  ( <parameter list>} is
     format
         format definition
     end format;
   [ activate
        <activation part>
     end activate;]
end screen;
```

Figure 1.2: Definition of Screen Type in AdaRel

how large volumes of data can easily be accomodated, while at the same time the user can interact with the screens choosing what he wishes to see. Complete definitions of the language extension plus examples was reported on in [HighLevelIO]. Finally, ideas about future directions of app'ication generators were presented in [APPIDEAS].

1.3 Office Information Systems

This work was undertaken between the principal investigator and Balaji Narasimhian. By studying the programming needs of offices, which are often data intensive, we hoped to be led towards new programming language facilities that support and enhance the database interface described previously. From this study we concluded that there was a major area of software development which is inadequately supported by current programming languages. This is the area of software that interacts with users, the user interface. Programming user interface applications is becoming a standard activity and yet, conventional programming languages have operators that deal with characters and lines and not with screens or sequences of screens. A second point of inspiration that resulted from the study is that the most common form of paper-user interaction is with a form. Therefore we concluded that a computer-based form seems a natual basis for an office information system. Pursuing such an environment, we have defined the basic properties of a form and the operations that must be supported by a forms-based system. These include: 1. form template definition; 2. form template instantiation; 3. specifying actions on form instances such as mailing, copying, saving and triggering; 4. validation of forms; and 5. storage and retrieval of forms and their contents. This work was summarized in the paper *The Design of Office Information Systems*, [OIS].

1.4 Software Engineering

This area of study was done jointly with Ronald Williamson and the principal investigator. In its most general terms it is concerned with improving the overall productivity of the software development process. The approach is to try and capture all of the elements of this process as it is being produced and to place them in a database management system in a uniform way. This permits one to query the database of elements and to answer questions that concern these elements over the entire software life cycle.

Other researchers have taken the approach of defining a formal language into which requirements and design can be phrased. Our belief was that such formal methods, though attractive from the point of view of tool creation, expected too much from the programmer. Our work captures the life-cycle information *without* requiring knowledge of a special language. The developer enters his text as if it were an editor and only points to key elements. These are then automatically translated into data items for the dbms. By capturing the data in a way that requires little or no additional effort, we believe that our approach will be practical. A second aspect of our design was theoretical. We had to model the information in the database. This was done in terms of a graph model that has a special structure, namely a collection of trees with cross connections. Using the formal model, we then defined abstractly basic notions for retrieval of information across elements of the life-cycle.

Our design was followed by an actual prototype that was built and is running on a Xerox 1100 under the SMALLTALK environment. The use of the system is summarized in [SODOS:USE]. The definition of the programming concepts and the use of an object-based methodology is given in [SODOS:Definition]. The actual implementation is discussed in [SODOS:Implementation].

1.5 SCriptWriter

The path of true love and research is often not straight. To understand the SCriptWriter Software Development Environment, it is useful to consider how we arrived at such a project. We were led to *SCriptwriter* from two directions: one being our research on Application Generators and the other being the particular computer systems we purchased. From the Application Generator work we realized that better systems could only be built if the domain of application was well focused. A second conclusion was that there was a great need for tools to design systems that involve interactive screens of information. Our second influence was the work being done on our IBM PC/ATs and the observation that they represented a delivery system upon which computer-based instruction could be run for large numbers of people. Thus we set ourselves the goal of devising a software development environment that would be suitable for the task of producing interactive, screen-based applications on microcomputers.

SCriptWriter is a software development environment that supports the development of multi-media productions. Its hardware consists of a dual monitor IBM PC/AT attached to a digitizer, sound chip and laser disk.

The software consists of 4 basic components: a command interface, a disk manager, an object-based programming language and a collection of editors. The editors are available for handling text, graphics, animation, laser disk, and font definition.

Though I cannot discuss all of the features encompassed in the system in this summary, I will point to two main features. One is the metaphor that is used. Creating a production can be quite complicated. The author needs some mental model so he can be guided as he creates his production. The metaphor employed in *ScriptWriter* is that of a play. Just as a play has actors, scenery, director and stagehands, so does *ScriptWriter*. A second major feature is the *IQ* programming language. This is an object-based programming language that includes the notion of player and lines as high-level concepts in the language. One goal we have attempted to achieve is to make the environment commands consistent with the language. By this I mean that all operations that can be performed at the environment level can also be performed in the programming language.

Over the past year of the grant we have designed the *SCriptWriter* system and begun its development. It is the activity that has dominated our time and collectively we are all quite excited about the research. The current design of the system is described in [SC REF]. A study of the design of the user interface is discussed in [SC UI].

Chapter 2

Staff

The following people have been supported on this grant during its duration.

Horowitz, Ellis	Principal investigator
Kemper, Alfons	Graduated 8-1-84 with his Ph.D. Currently Research Assistant Professor at Univeristy of Karlsruhe
Williamson, Ronald	Graduated 10-1-84 with his Ph.D. Now Research Scientist at Hughes Aircraft Corp.
Narasimhan, Balaji	Currently a graduate student in Computer Science
Papa, Marco	Currently a graduate student in Computer Science. working on his Ph.D. thesis. Working title is A Performance Model for Real Time Computer Animation Expected graduation June 1987.
Bills, Mark	currently working on the development of the IQ language, a part of SCriptWriter, while working on a B.S. degree in Computer Science. Graduated with M.S. program in June '86.
Anderson, David	Currently working on the development of the IQ language, a part of SCriptWriter. Graduated with B.S. degree in June '86.
Garg, Pankaj	Graduate student in Computer Science just starting to work towards his Ph.D. degree

Chapter 3

Publications

AppGenSur "A Survey of Application Generators", *IEEE Software*, vol. 2 No. 1, Jan. 1985, 40-54 (with Kemper and Narasimhan)

AdaRel "AdaRel: A Relational Extension of Ada", Computer Science Technical Report, U.S.C., October 1983.

HighLevelIO "High-Level Input/Output Facilities in a Database Programming Language", *Proc. International Conf. on System Sciences*, Jan. 1985, 67-80. (with A. Kemper)

APPIDEAS "Application Generators: Ideas for Programming Language Extensions", *Proc. ACM Annual Conf.*, San Francisco, Oct. 8-10, 1984, 94 - 101. (with A. Kemper and B. Narasimhan).

OIS "The Design of Office Information Systems", Technical Report TR-83-320, Computer Science Dept., U.S.C. December, 1983.

SODOS: Definition "SODOS Software Documentation Support Environment - Its Definition" *IEEE Trans. on Software Engineering*, vol. no. 1986, (with R. Williamson)

SODOS: Use "SODOS Software Documentation Support Environment - Its Use" *IEEE Trans. on Software Engineering*, vol. no. 1986, (with R. Williamson)

SODOS: Implementation "SODOS A Software Documentation Support Environment: Its Implementation", Computer Science Dept., U.S.C. Sept. 1984

SODOS:Summary "SODOS A Software Documentation Support Environment: Summary", *Proc. IEEE Software Engineering Conference*, London, Aug. 1985

SC REF "Scriptwriter Reference Manual" Computer Science Dept., U.S.C. April, 1985.

SC-UI "Design Considerations for Hierarchical, Keyword-Based Menu Interfaces", Computer Science Dept., U.S.C. May, 1985.

END

8-87

DTIC